

In the claims:

For the Examiner's convenience, all pending claims are presented below with changes shown in accordance with the mandatory amendment format.

1. (Currently Amended) A computer-implemented method comprising:

monitoring thread switches in a multiple-threaded application through use of a single thread switch flag;

executing a non-blocking thread synchronization sequence; and

interrupting the non-blocking thread synchronization sequence upon the occurrence of a thread switch; and

repeating the non-blocking thread synchronization sequence, wherein the non-blocking thread synchronization sequence is idempotent in order to abandon the non-blocking thread synchronization sequence in mid-sequence without consequences.
2. (Cancelled)
3. (Currently Amended) The computer-implemented method of claim 1 ~~[[2]]~~ wherein the multiple-threaded applications are supported by a computer programming language selected from the group consisting of JAVA, C#, ~~CLI~~, LISP, and Pascal.
4. (Cancelled)
5. (Currently Amended) The computer-implemented method of claim 1 ~~[[2]]~~ wherein the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.

6. (Currently Amended) The computer-implemented method of claim 5 wherein executing the frontier pointer-based allocation sequence comprises:

loading a frontier pointer into a first register;

moving a current value of the frontier pointer to a second register;

adding the size of an object to be allocated to the first register such that a new frontier pointer is determined;

storing a virtual method table to the second register if a thread switch has not occurred; and

updating the frontier pointer with the new frontier pointer if a thread switch has not occurred.

7. (Currently Amended) A machine-readable medium that provides executable instructions, which when executed by a processor, cause the processor to perform a method, the method comprising:

monitoring thread switches in a multiple-threaded application through use of a single thread switch flag;

executing a non-blocking thread synchronization sequence; and

interrupting the non-blocking thread synchronization sequence upon the occurrence of a thread switch; and

repeating the non-blocking thread synchronization sequence, wherein the non-blocking thread synchronization sequence is idempotent in order to abandon the non-blocking thread synchronization sequence in mid-sequence without consequences.

8. (Cancelled)

9. (Currently Amended) The machine-readable medium of claim 7 [[8]] wherein the multiple-threaded applications are supported by a computer programming language selected from the group consisting of JAVA, C#, ~~CLI~~, LISP, and Pascal.

10. (Cancelled)

11. (Currently Amended) The machine-readable medium of claim 7 [[8]] wherein the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.

12. (Original) The machine-readable medium of claim 11 wherein executing the frontier pointer-based allocation sequence comprises:

loading a frontier pointer into a first register;

moving a current value of the frontier pointer to a second register;

adding the size of an object to be allocated to the first register such that a new frontier pointer is determined;

storing a virtual method table to the second register if a thread switch has not occurred; and

updating the frontier pointer with the new frontier pointer if a thread switch has not occurred.

13. (Currently Amended) A computing system comprising:

at least one central processing unit, the central processing unit executing multi-threaded applications;

a thread switch indicator to indicate the occurrence of a thread switch; and

an instruction set to implement non-blocking thread synchronization sequences such that partially completed non-blocking thread synchronization sequences used to share resources local to the at least one central processing unit can be abandoned and repeated upon the occurrence of a thread switch;

wherein the non-blocking thread synchronization sequences are idempotent in order to abandon the non-blocking thread synchronization sequences in mid-sequence without consequences.

14. (Original) The computing system of claim 13 wherein the instruction set includes:

a set instruction to set the thread switch indicator upon the occurrence of a thread switch;

a first conditional move instruction to move data if the thread switch indicator is set;

a second conditional move instruction to move data if the thread switch indicator is not set;

a first jump instruction to bypass instructions if the thread switch indicator is set;

a second jump instruction to bypass instructions if the thread switch indicator is not set; and

a clear instruction to clear the thread switch indicator.

15. (Original) The computing system of claim 14 wherein the thread switch indicator is a thread switch flag.

16. (Original) The computing system of claim 13 wherein each of the at least one central processing units has a single allocation area and the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.

17. (Currently Amended) The computing system of claim 13, wherein the computing system uses a computer programming language selected from the group consisting of JAVA, C#, ~~CLI~~, LISP, and Pascal.

18. (Currently Amended) A computer system executable instruction set comprising:

a thread switch indicator to indicate the occurrence of a thread switch;

a set instruction to set the thread switch indicator upon the occurrence of a thread switch;

a first conditional move instruction to move data if the thread switch indicator is set;

a second conditional move instruction to move data if the thread switch indicator is not set;

a first jump instruction to bypass instructions if the thread switch indicator is set;

a second jump instruction to bypass instructions if the thread switch indicator is not set; and

a clear instruction to clear the thread switch indicator;

wherein a non-blocking thread synchronization sequence is abandoned and repeated upon the occurrence of a thread switch as indicated by the thread switch indicator, the non-blocking thread synchronization sequence being idempotent in order to abandon the non-blocking thread synchronization sequences in mid-sequence without consequences.

19. (Currently Amended) The computer system instruction set of claim 18 implemented as hardware.

20. (Currently Amended) The computer system executable instruction set of claim 18 wherein the thread switch indicator is a thread switch flag.

21. (Currently Amended) The computer system executable instruction set of claim 18 used to implement a non-blocking thread synchronization sequence for the execution of multi-threaded applications.

22. (Currently Amended) The computer system executable instruction set of claim 21 wherein the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.